2-D Neighborhoods

- 4-point neighborhood

$$
\begin{array}{ccc}
 & \circ & \\
\circ & \times & \circ \\
 & \circ &
\end{array}
$$

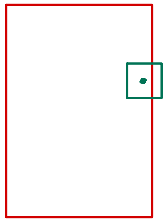
$$\partial(i,j) = \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$$

- 8-point neighborhood

$$
\begin{array}{ccc}
\circ & \circ & \circ \\
\circ & \times & \circ \\
\circ & \circ & \circ
\end{array}
$$

$$\partial(i,j) = \left\{ \begin{array}{c} (i+1,j),(i-1,j),(i,j+1),(i,j-1) \\ (i+1,j+1),(i-1,j+1) \\ (i+1,j-1),(i-1,j-1) \end{array} \right\}$$

- More generally, a *Neighborhood System* is any mapping with the two properties that:
  1. For all $s \in S$, $s \notin \partial s$
  2. For all $r \in S$, $r \in \partial s \Rightarrow s \in \partial r$



Boundary Conditions

- How do you process pixels on the boundary of an image??
- Consider the following example using a 4-point neighborhood

A small example image

$$\begin{array}{cccc} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{array}$$

4 3 2 1 1 2 3 4
4 3 2 1 2 3 4

4 neighbors of l

$$\begin{array}{ccc} & l_1 & \\ l_4 & l & l_2 \\ & l_3 & \end{array}$$

- Free boundary condition - "same", aka zero padding

$$\partial l = \{h, p, k\}$$
$$\partial p = \{l, o\}$$

video game

- Toriodal boundary condition (asteroids) - "wrap around"

FFT assumes this B.C. type
aka DFT (discrete Fourier transform)

$$\partial l = \{h, i, p, k\}$$
$$\partial p = \{l, m, d, o\}$$

- Reflective boundary condition

$$l_1 = h,\ l_2 = k,\ l_3 = p,\ l_4 = k$$
$$p_1 = l,\ p_2 = o,\ p_3 = l,\ p_4 = o$$

DCT - discrete cosine transform

Edge Detection

- Edges
  - Edges naturally occur in images due to the discontinuities form by occlusion.
  - Edges often delineate the boundaries between distinct regions.
  - Edges often contain important visual and semantic information.
- Edge detection:
  - The process of identifying pixels that fall along edges.
  - As width any detect process subject to a trade-off between false alarm and missed detection rates.
- Performance Metrics:
  - Evaluation of edge detection scemes can be difficult.
  - Correct labeling of edge and non-edge pixels often requires subjective interpretation.
  - Best choice of edge detection scheme usually depends on task.
  - Performance metrics exist and usually use synthetic data input for evaluation.

Gradient Based Edge Detection

- Compute local estimate of gradient

$$\nabla \underbrace{f(x,y)}_{\text{image}} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- From these, compute gradient magnetude and angle.

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

- Apply threshold to the magnitude of gradient

$$\left.\begin{array}{rl} \text{edge} & |\nabla f| \geq T \\ \text{no edge} & |\nabla f| < T \end{array}\right\} \text{tradeoff}$$

- Choosing T
  - Too large $\Rightarrow$ missed detections
  - Too small $\Rightarrow$ false alarms

Can be tested on this ☆

How to Compute Gradient

impulse responses ≡ point spread functions

- Directional derivatives can be computed by applying a spatial filter. Impulse responses of the filters that compute the gradients (these are called kernels)
- Conventional (off center)

will cause edge estimate to be shifted by ½ pixel

$$\begin{bmatrix} \boxed{-1} & 1 \\ 0 & 0 \end{bmatrix} \quad \begin{bmatrix} \boxed{-1} & 0 \\ 1 & 0 \end{bmatrix}$$

- Roberts (off center)

$$\begin{bmatrix} \boxed{0} & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} \boxed{1} & 0 \\ 0 & -1 \end{bmatrix}$$

- Prewitt (on center)

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \boxed{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

- Sobel (on center)

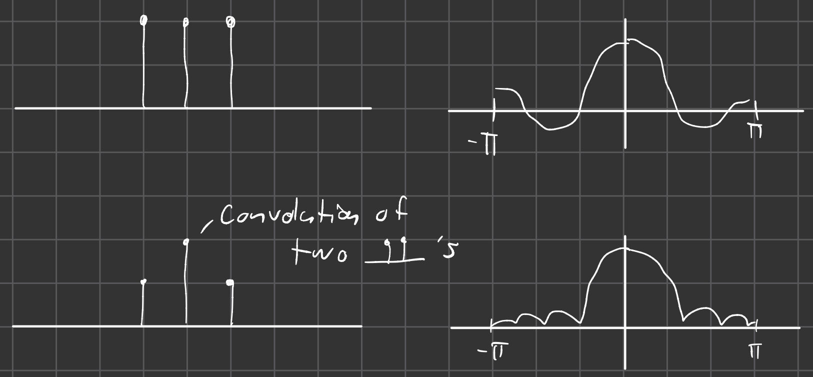
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

adapt threshold locally

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

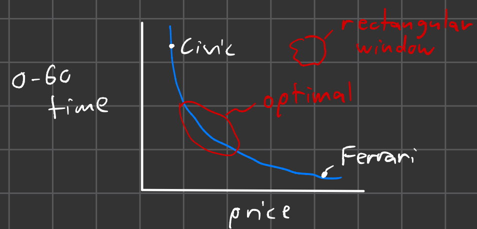
psinc?

Canny edge detector ☆



?

window with a rect → huge side lobes
Hamming window



DC gain is sum of impulse responses (discrete area under curve)

Second Derivative Edge Detection → See notes on website

$f(n)$ ← time

$$\nabla f(n+\tfrac{1}{2}) = \frac{f(n+1)-f(n)}{1}$$

$$\nabla f(n-\tfrac{1}{2}) = \frac{f(n)-f(n-1)}{1}$$

$$\frac{df}{dn} =$$

$$\frac{d^2f}{dn} = \frac{[f(n+1)-f(n)]-[f(n)-f(n-1)]}{1} = -2\left(f(n) - \tfrac{1}{2}\left(f(n-1)+f(n+1)\right)\right)$$

$$\nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = -4\left(f(m,n) - \langle f(m,n)\rangle\right)$$

$$= \frac{1}{4}\begin{pmatrix} f(m-1,n) + f(m,n-1) \\ + f(m+1,n) + f(m,n+1) \end{pmatrix}$$

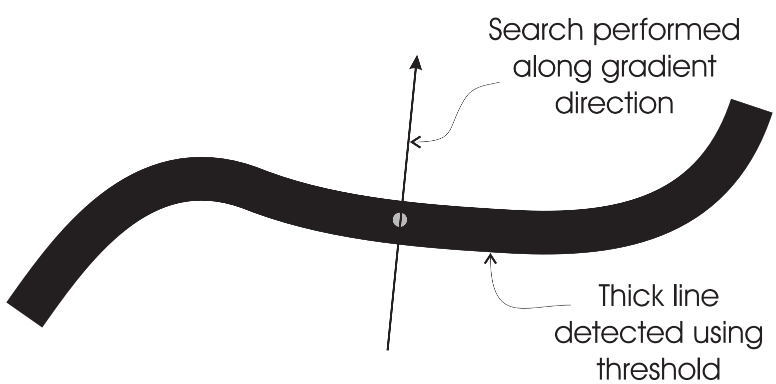
Laplacian

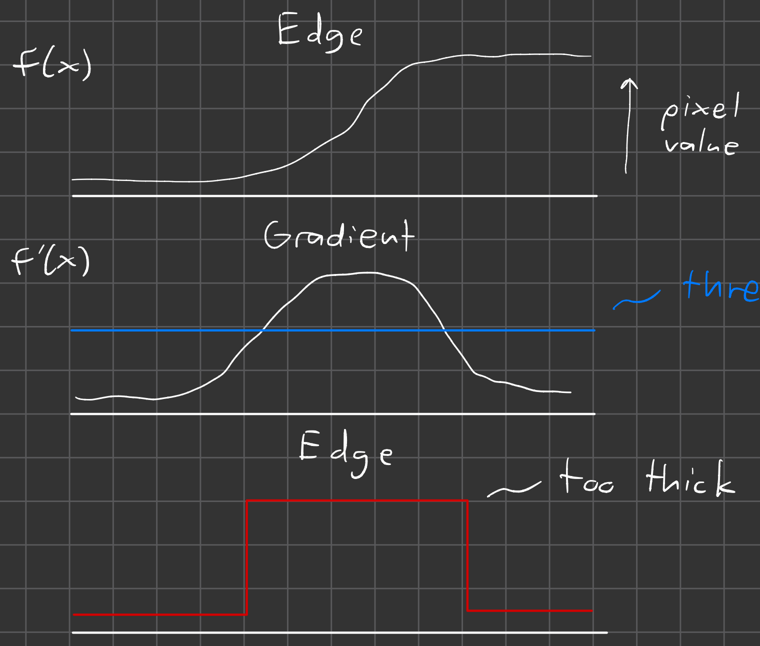
pixels minus their local average

Edge Thining

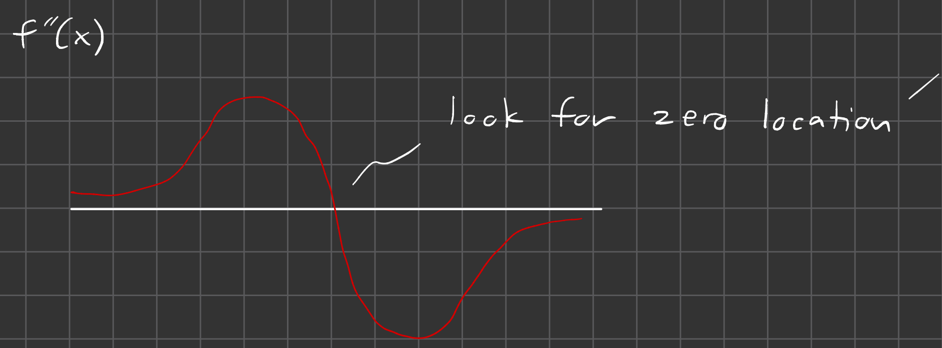
- Thresholding of gradient magnitude generally produces a thick edge.
- Edge should be thinned to produce most accurate result.

1. Set $S = \{s : |\nabla f(s)| \geq T\}$
2. Set $D = \emptyset$ (detected edge points)
3. For each $s \in S$
   (a) Compute $\theta =$ gradient direction at s.
   (b) Select out $P =$ all pixels in direction θ starting at s within maximum distance d_{max} from s.
   (c) If $|\nabla f(s)| \geq \max_{p \in P} \{|\nabla f(p)|\}$, then
$$D \leftarrow D + \{s\}$$





First Check



Second Check

In 2D, you get the Laplacian